Solving SAT with Active Membranes and Pre-Computed Initial Configurations

Bogdan Aman, Gabriel Ciobanu

Romanian Academy, Institute of Computer Science
Blvd. Carol I no.11, 700506 Iaşi, Romania
baman@iit.tulasi.ro, gabriel@info.uaic.ro

Summary. In this paper we provide algorithms for solving the SAT problem using P systems with active membranes with neither polarization nor division rules. The semi-uniform solutions are given under the assumption that initial configurations (either alphabet or structure) of exponential size are pre-computed by well-defined P systems (P systems with replicated rewriting and P systems with active membranes and membrane creation, respectively) working in polynomial time. An important observation is that we specify how the pre-computed initial configurations are constructed.

1 Introduction

Membrane computing is inspired by the architecture and the behaviour of living cells. Various classes of membrane systems (also called P systems) have been defined in [9], while several applications of these systems are described in [3]. Membrane systems are characterised by three features: (i) a membrane structure consisting of a hierarchy of membranes (which are either disjoint or nested), with an unique top membrane called the skin; (ii) multisets of objects associated with membranes; (iii) rules for processing the objects and membranes. When membrane systems are seen as computing devices, two main research directions are usually considered: computational power in terms of the classical notion of Turing computability (e.g., [1]), and efficiency in algorithmically solving NP-complete problems in polynomial time (e.g., [2]). Thus, membrane systems define classes of computing devices which are both powerful and efficient.

Under the assumption that $P \neq NP$, efficient solutions to NP-complete problems cannot be obtained without introducing features which enhance the efficiency of the system ways to exponentially grow the workspace during the computation, nondeterminism, and so on). For instance, some pre-computed resources are used in [4, 6].

In this paper we consider P systems with active membranes [7], and show that they can provide simple semi-uniform solutions to the SAT problem without using neither polarization nor division, but using exponential size pre-computed
initial configurations (either alphabet or structure). An important observation is that we specify how our pre-computed initial configurations are constructed in a polynomial number of steps by additional well-defined P systems (P systems with replicated rewriting and P systems with active membranes and membrane creation, respectively).

2 Preliminaries

We consider polarizationless P systems with active membranes [7]. The original definition also includes division rules, rules that are not needed here.

**Definition 1.** A polarizationless P system with active membranes is a tuple 
\[ \Pi = (\Gamma, \Lambda, \mu, w_1, \ldots, w_d, R) \], where:

- \( d \geq 1 \) is the initial degree;
- \( \Gamma \) is a finite non-empty alphabet of objects;
- \( \Lambda \) is a finite set of labels for membranes;
- \( \mu \) is a membrane structure (i.e., a rooted unordered tree, usually represented by nested brackets) in which each membrane is labelled by an element of \( \Lambda \) in a one-to-one way;
- \( w_1, \ldots, w_d \) are strings over \( \Gamma \), describing the initial multisets of objects placed in a number of \( d \) membranes of \( \mu \);
- \( R \) is a finite set of rules over \( \Gamma \):
  1. \([a \rightarrow w]_h\) object evolution rules
     An object \( a \) is rewritten into the multiset \( w \), if \( a \) is placed inside a membrane labelled by \( h \).
  2. \([a]_h \rightarrow [b]_h\) send-in communication rules
     An object \( a \) is sent into a membrane labelled by \( h \), becoming \( b \).
  3. \([a]_h \rightarrow [b]_h\) send-out communication rules
     An object \( a \), placed into a membrane labelled by \( h \), is sent out of membrane \( h \) and becomes \( b \).
  4. \([a]_h \rightarrow b\) dissolution rules
     A membrane \( h \) containing an object \( a \) is dissolved, while object \( a \) is rewritten to \( b \).

Each configuration \( C_i \) of a P system with active membranes and input objects is described by the membrane structure, together with the multisets of objects located in the corresponding membranes. The initial configuration of such a system is denoted by \( C_0 \). An evolution step \( C_i \Rightarrow C_{i+1} \) from a configuration \( C_i \) to a new configuration \( C_{i+1} \) is done according to the following principles:

- Each object is involved in at most one rule per step, while each membrane could be involved in several rules.
- The application of rules is maximally parallel: all rules that can be applied are applied.
• When several conflicting rules could be applied at the same time, a nondeterministic choice is performed; this implies that multiple configurations can be reached as the result of an evolution step.
• In each evolution step, all evolution rules are applied inside the most inner membranes, followed by all communication rules involving the membranes themselves. This process is then repeated to the membranes containing them, and so on towards the skin membrane.
• Objects sent out from the skin membrane represent the computation result.

A halting evolution of such a system $\Pi$ is a finite sequence of configurations $\vec{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$, such that $\mathcal{C}_0 \Rightarrow \mathcal{C}_1 \Rightarrow \ldots \Rightarrow \mathcal{C}_k$, and no rules can be applied any more in $\mathcal{C}_k$. A non-halting evolution $\vec{C} = (\mathcal{C}_i | i \in \mathbb{N})$ consists of infinite evolution $\mathcal{C}_0 \Rightarrow \mathcal{C}_1 \Rightarrow \ldots$, where the applicable rules are never exhausted.

3 Solving the SAT Problem with Active Membranes

At the beginning of 2005, Gh. Păun wrote:

“My favourite question (related to complexity aspects in P systems with active membranes and with electrical charges) is that about the number of polarizations. Can the polarizations be completely avoided? The feeling is that this is not possible - and such a result would be rather sound: passing from no polarization to two polarizations amounts to passing from non-efficiency to efficiency.”

This conjecture (problem F in [8]) can be formally described in terms of membrane computing complexity classes as follows:

$$P = PMC_{\mathcal{AM}^\mathcal{R} (+d, -n, +e, +c)}$$

where

• $PMC_{\mathcal{R}}$ indicates that the result holds for P systems with input membrane;
• $+d$ indicates that dissolution rules are permitted;
• $-n$ indicates that only division rules for elementary membranes are allowed;
• $+e$ indicates that evolution rules are permitted;
• $+c$ indicates that communication rules are permitted.

The SAT problem checks the satisfiability of a propositional logic formula in conjunctive normal form (CNF). Let $\{x_1, x_2, \ldots, x_n\}$ be a set of propositional variables. A formula in CNF is of the form $\varphi = C_1 \land C_2 \land \cdots \land C_m$ where each $C_i$, $1 \leq i \leq m$ is a disjunction of the form $C_i = y_1 \lor y_2 \lor \cdots \lor y_r$ ($r \leq n$), where each $y_j$ is either a variable $x_k$ or its negation $\neg x_k$.

We present some attempts to solve this conjecture by providing algorithms solving the SAT problem using P systems with active membranes with neither polarizations nor division, but using exponential pre-computed initial configurations constructed by additional P systems in polynomial time.
3.1 Solving SAT Problem by Using a Pre-Computed Alphabet

In this section, we propose a polynomial time solution to the SAT problem using the polarizationless P systems with active membranes, without division, but with a pre-computed alphabet. For any instance of SAT we construct effectively a system of membranes that solves it. Formally, we prove the following result:

**Theorem 1.** The SAT problem can be solved by a polarizationless P system with active membranes and **without division**, but with an exponential alphabet pre-computed in linear time with respect to the number of variables and the number of clauses, i.e.,

\[
P = PMC_{\mathcal{A}M^0(+d, t, \lambda, \text{pre}(\alpha))},
\]

where \(\text{pre}(\alpha)\) indicates that a pre-computed alphabet is permitted.

**Proof.** Let us consider a propositional formula

\[\varphi = C_1 \land C_2 \land \cdots \land C_m\]

where each \(C_i, 1 \leq i \leq m\) is a disjunction of the form

\[C_i = y_1 \lor y_2 \lor \cdots \lor y_r\ (r \leq n),\]

where each \(y_j\) is either a variable \(x_k\) or its negation \(\neg x_k\).

We construct a P system with active membranes able to check the satisfiability of \(\varphi\). The P system is given by \(\Pi = (\Gamma, \Lambda, \mu, w_1, \ldots, w_d, R)\), where:

- \(V = \{z_i \mid 0 \leq i \leq \max\{m, n\}\} \cup \{s_i \mid i = t_1 \ldots t_n, t_j \in \{0, 1\} \text{ and } 1 \leq j \leq n\} \cup \{yes, no\}\).

The alphabet \(\{s_i \mid i = t_1 \ldots t_n, t_j \in \{0, 1\} \text{ and } 1 \leq j \leq n\}\) to be placed inside the input membrane 0 can be generated, starting from an object \(s\), using the rules:

- \(s \rightarrow s_0 s_1;\)
- \(s_i \rightarrow s_0 s_{i1}, \text{ for } i = t_1 \ldots t_k\) where \(t_j \in \{0, 1\} \text{ and } 1 \leq j \leq k < n.\)

Thus all the possible assignments for the \(n\) variable \(\{x_1, x_2, \ldots, x_n\}\) are created. The rules are applied until the length \(k\) of \(i\) in the second rule equals \(n\). For example, \(s_{100}\) over \(\{x_1, x_2, x_3\}\) represents the assignment \(x_1 = 1, x_2 = 0\) and \(x_3 = 0\) (1 stands for true, while 0 stands for false). The input alphabet can be computed in linear (polynomial) time by using an additional device, for instance P systems with replicated rewriting [5].

- \(A = \{0, c_1, \ldots, c_m, h\}\), with \(c_i = z_1 \ldots z_n, 1 \leq i \leq m\) where
  - \(z_j = 1\) if \(x_j\) appears in \(C_i\);
  - \(z_j = 0\) if \(\neg x_j\) appears in \(C_i\);
  - \(z_j = \star\) if neither \(x_j\) nor \(\neg x_j\) appear in \(C_i\).

For example \(c_1 = 1 \star 0\) over the set of variables \(\{x_1, x_2, x_3\}\) represents the disjunction \(c_1 = x_1 \lor \neg x_3.\)

- \(\mu = [[[[[0]c_1]c_2 \cdots c_{m-1}]c_m]h.\)
- \(w_0 = z_0.\)
- \(w_i = \lambda, \text{ for all } i \in A \setminus \{0\}.\)
- The set \(R\) contains the following rules:
1. $[z_0]_0 \rightarrow z_0$
After the input is placed inside membrane 0, membrane 0 is dissolved, and its content is released in the upper membrane labelled with $c_1$.

2. $[s_i]_{c_j} \rightarrow s_i[ ]_{c_j}$
if $i$ and $j$ have at least one position with the same value (either 0 or 1);
$[s_i]_{c_m} \rightarrow yes$
if $i$ and $m$ have at least one position with the same value (either 0 or 1).

An assignment $s_i$ is sent out of a membrane $c_m$ if there is at least one position in $i$ and $j$ that is equal, namely an assignment to a variable $x_k$ such that it makes $C_j$ true. Once an object $yes$ is generated, another object $yes$ cannot be created because membrane $c_m$ was dissolved and the rule $[s_i]_{c_m} \rightarrow yes$ cannot be applied. For example, if $c_1 = 1 \ast 0$ and $s_{101}$ (as described above), then this means that $s_{101}$ satisfies the clause coded by $c_1 = 1 \ast 0$ since both have 1 on their first position, and this is enough to make true a disjunction.

3. $[z_i]_{c_1} \rightarrow [ ]_{c_i} z_{i+1}$, for $1 \leq i \leq m - 1$
$[z_m]_{c_m} \rightarrow no$
The object $z_0$ waits a step after membrane 0 is dissolved in order to allow the other objects $s_i$ to go through the $c_j$ membranes. The object $z_i$ then is communicated through the $c_j$ membranes. Once $z_m$ reached the membrane $c_m$, if membrane $c_m$ still exists (i.e., the rule $[s_i]_{c_m} \rightarrow yes$ was not applied), then the answer $no$ is generated. Once an object $yes$ or $no$ is generated, other objects $yes$ or $no$ cannot be created because membrane $c_m$ was dissolved, and neither rule $[s_i]_{c_m} \rightarrow yes$ nor $[z_m]_{c_m} \rightarrow no$ can be applied.

4. $[yes]_h \rightarrow yes[ ]_h$
$[no]_h \rightarrow no[ ]_h$
The answer $yes$ or $no$ regarding the satisfiability is sent out of the skin.

3.2 Solving SAT Problem Using a Pre-Computed Initial Structure

In this section, we propose a polynomial time solution to the SAT problem using the polarizationless P systems with active membranes and without division, but with a pre-computed structure. For any instance of SAT we construct effectively a system of membranes that solves it. We also enforce another principle needed to perform an evolution step: each membrane can be subject to at most one communication rule per step. This principle is needed when generating all possible assignments to be verified. Formally, we prove the following result:

**Theorem 2.** The SAT problem can be solved by a polarizationless P system with active membranes and without division, but with an initial exponential structure pre-computed in linear time with respect to the number of variables and the number of clauses, i.e.,

$$ P = PMC_{AM^{pre(\mu)}}(+d,+e,+c,pre(\mu)) $$

where $pre(\mu)$ indicates that a pre-computed structure is permitted.
Proof. Let us consider a propositional formula
\[ \varphi = C_1 \land C_2 \land \cdots \land C_m \]
where each \( C_i, 1 \leq i \leq m \) is a disjunction of the form
\[ C_i = y_1 \lor y_2 \lor \cdots \lor y_r \quad (r \leq n), \]
where each \( y_j \) is either a variable \( x_k \) or its negation \( \neg x_k \).

We construct a P system with active membranes able to check the satisfiability of \( \varphi \). The P system is given by \( \Pi = (\Gamma, \Lambda, \mu, w_0, \ldots, w_d, R) \), where:

- \( V = \{a_i, t_i, t'_i, f_i, f'_i \mid 1 \leq i \leq n\} \cup \{z_i \mid 0 \leq i \leq 4 \times n + 2 \times m\} \cup \{\text{yes, no}\} \).
- \( \Lambda = \{0, \ldots, n, c_1, \ldots, c_m, h\}, 1 \leq i \leq m \).
- \( \mu = \left[\begin{array}{l}
\ldots \\
2 \\
1 \\
0 \\
h
\end{array}\right] \), where
  - each membrane \( i \) contains two membranes \( i + 1 \) for \( 0 \leq i \leq n - 1 \);
  - each membrane \( n \) contains a membrane structure \( \left[\begin{array}{l}
\cdots [c_m \cdots c_l]c_a
\end{array}\right] \);
  - membrane 0 is the input membrane.

Graphically, the membrane structure \( \mu \) can be represented as a tree:

This membrane structure can be generate in linear (polynomial) time with respect to the number of variables and the number of clauses. This is done by using an additional device that starts from a membrane structure \( \left[\begin{array}{l}
\cdots [c_0]h
\end{array}\right] \), with object 0 placed inside membrane 0 and rules of the form:

- \( [i \to (i + 1)' (i + 1)']_0 \), for \( 0 \leq i \leq n - 1 \)
- \( i' \to [i]_i \), for \( 1 \leq i \leq n \)
- \( n \to [c_2]_1 \)
- \( c_k \to [c_{k+1}]_{c_k} \), for \( 2 \leq k \leq m - 1 \)
- \( c_m \to [c_{m+1}]_{c_m} \).

- \( w_0 = a_1z_0 \).
- \( w_i = \lambda \), for all \( i \in \Lambda \setminus \{0\} \).

The set \( R \) contains the following rules:

1. \( [z_i \to z_{i+1}]_0 \), for all \( 0 \leq i < 4 \times n + 2 \times m \)

   These rules count the time needed for producing the truth assignments for the \( n \) variables inside the membranes labelled by \( n \) \((3 \times n)\) steps, then to
solve the membranes labelled by \( c_j \), \( 1 \leq j \leq m \) (2 × m steps), and for an \( y \) object to reach the membrane labelled by 0 (0 steps).

2. \([a_i \rightarrow t_i, f_i]_{i-1}, \) for \( 1 \leq i \leq n \)
   \( t_i \) \( \rightarrow [t_i], \) for \( 1 \leq i \leq n \)
   \( f_i \) \( \rightarrow [f_i], \) for \( 1 \leq i \leq n \)
   \( t_i \rightarrow t_i'_{a_i+1}], \) for \( 1 \leq i \leq n - 1 \)
   \( t_i' \) \( \rightarrow [t_i], \) for \( i + 1 \leq k \leq n \)
   \( t_i \rightarrow t_i'_{a_i+1}], \) for \( 1 \leq i \leq n - 1 \)
   \( f_i \rightarrow f_i'_{a_i+1}], \) for \( 1 \leq i \leq n - 1 \)
   \( f_i' \) \( \rightarrow [f_i], \) for \( i + 1 \leq k \leq n \)
   \( f_i \rightarrow f_i'_{a_i+1}], \) for \( 1 \leq i \leq n - 1 \)

In membranes \( n \) we create all possible assignments for the \( n \) variable \( \{x_1, x_2, \ldots, x_n\} \). It starts from an object \( a_1 \) placed initially in membrane labelled by 0. Each \( a_i \) is used to create \( t_i \) and \( f_i \) that are then send in one of the two membranes labelled by \( i \) placed in membrane \( i - 1 \). In fact each membrane \( i \) receives either \( t_i \) or \( f_i \), and this is possible because a membrane can be involved in only one communication rule of an evolution step. After an object \( t_i \) or \( f_i \) reaches a membrane \( i \), it generates two new copies of it to be sent inside membranes \( i + 1 \) together with an object \( a_i+1 \) that is used then to construct the assignments over variable \( x_{i+1} \).

3. \( t_i \) \( \rightarrow [t_i] \), if \( x_i \) appears in \( C_j \)
   \( t_i \) \( \rightarrow [t_i], \) for \( 1 \leq i \leq n \)
   \( [t_i]_{e_m} \rightarrow y, \) for \( 1 \leq i \leq n \)
   \( f_i \) \( \rightarrow [t_i] \), if \( \neg x_i \) appears in \( C_j \)
   \( [f_i]_{e_j} \rightarrow f_i, \) for \( 1 \leq i \leq n \)
   \( [f_i]_{e_m} \rightarrow y, \) for \( 1 \leq i \leq n. \)

An assignment \( t_i \) (\( f_i \)) is sent into a membrane \( c_j \) if there is an assignment to a variable \( x_k \) (\( \neg x_k \)) such that it makes \( C_k \) true. Once all membranes labelled by \( c_i \) are dissolved inside a membrane labelled by \( n \), an object \( y \) is generated.

4. \( [y] \rightarrow [y] \), for \( k \in A \setminus \{0, h\} \)
   \( [y]_{0} \rightarrow yes \)
   \( [z_{4×n+2×m}]_{0} \rightarrow no. \)

The object \( z_0 \) waits for \( 4 \times n + 2 \times m \) steps in order to allow dissolving the membrane labelled by 0 if this still exists (i.e., the rule \( [y]_0 \rightarrow yes \) was not applied), then the answer \( no \) is generated. Once an object \( yes \) or \( no \) is generated, other objects \( yes \) or \( no \) cannot be created because membrane \( e_m \) was dissolved, and neither rule \( [y]_0 \rightarrow yes \) nor \( [z_{4×n+2×m}]_0 \rightarrow no \) can be applied.

5. \( [yes]_h \rightarrow yes \)
   \( [no]_h \rightarrow no. \)

The answer \( yes \) or \( no \) regarding the satisfiability is sent out of the skin.
Example 1. We illustrate this algorithm and the evolution of a system $II$ constructed for the propositional formula $\psi = (x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2)$.

Thus, $m=n=2$. The initial configuration of the systems, constructed by an additional device that starts from a membrane structure $[[[]]0]h$, with object 0 placed inside membrane 0 and rules of the form:

- $[0 \rightarrow 1', 1']_0$ and $[1 \rightarrow 2', 2']_1$
- $1' \rightarrow [1]_1$ and $2' \rightarrow [2]_2$
- $2 \rightarrow [c_2]_{c_1}$ and $c_2 \rightarrow [c_2]$.

The obtained structure is

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [1] [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]_0]_0]h$$

Graphically, the membrane structure $\mu$ can be represented as a tree:

Using the set $R$ of rules 1 ÷ 5, the computation proceeds as follows:

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [1] [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [1] [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]h$$

$$\Rightarrow$$

$$[[[[[ [c_2]_{c_1}]_2 [[[ [c_2]_{c_1}]_2 [2] [1] a_1 z_0]_0]_0]_0]_0]h$$

It can be noticed that even the object $z$ has now the subscript $4 \times n + 2 \times m = 4 \times 2 + 2 \times 2 = 12$, it cannot generate a no object because membrane labelled by 0 was already dissolved by an $y$ object in the previous step. Also, even another $y$ object reached the membrane labelled by 0, it cannot generate an yes object because membrane labelled by 0 was already dissolved by another $y$ object in a previous step.
4 Conclusion

In this paper we deal with a question presented by Páun in 2005: Can the polarizations be completely avoided? (related to complexity aspects of P systems with active membranes and with electrical charges). We answer positively to this question: we do not use polarizations in solving the SAT problem, but use a pre-computed initial configuration involving either exponential alphabet or exponential structure.

We proved $P = PMC_{AM^0(+d,+e,+c,pre(\alpha))}$ and $P = PMC_{AM^0(+d,+e,+c,pre(\mu))}$ by providing two algorithms for solving the SAT problem using polarizationless P system with active membranes and without division. For the former equality, the provided algorithm is using an exponential alphabet pre-computed in linear time by a P system with replicated rewriting, while the later one is using an initial exponential structure pre-computed in linear time with respect to the number of variables and the number of clauses by P systems with membrane creation.

References
